Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №3 по курсу «Моделирование»

Тема ОДУ второго порядка с краевыми условиями 2-го и 3-го рода

Студент Кононенко С.С.

Группа ИУ7-63Б

Оценка (баллы) ______________________________

Преподаватель Градов В.М.

Москва — 2021 г.

Тема работы

Программно-алгоритмическая реализация моделей на основе ОДУ второго порядка с краевыми условиями II и III рода.

Цель работы

Получение навыков разработки алгоритмов решения краевой задачи при реализации моделей, построенных на ОДУ второго порядка.

Теоретические сведения

Задана математическая модель:

$$\frac{d}{dx}(\lambda(T)\frac{dT}{dx}) - 4 \cdot k(T) \cdot n_p^2 \cdot \sigma \cdot (T^4 - T_0^4) = 0$$

Для модели заданы краевые условия:

$$\begin{cases} x = 0, -\lambda(T(0))\frac{dT}{dx} = F_0. \\ x = l, -\lambda(T(l))\frac{dT}{dx} = \alpha(T(l) - T_0) \end{cases}$$

Функции $\lambda(T)$ и $k(T)$ заданы таблицей (таблица приведена отдельно).

Заданы начальные параметры:

- $n_p = 1.4$ – коэффициент преломления;
- $l = 0.2$ см – толщина слоя;
- $T_0 = 300$K – температура окружающей среды;

- $\sigma = 5.668 \cdot 10^{-12}$ $\frac{\text{Вт}}{\text{см}^2 \cdot \text{К}^4}$ – постоянная Стефана–Больцмана;
- $F_0 = 100$ $\frac{\text{Вт}}{\text{см}^2}$ – поток тепла;
- $\alpha = 0.05$ $\frac{\text{Вт}}{\text{см}^2 \cdot \text{К}}$ – коэффициент теплоотдачи.

Выход из итераций организовать по температуре и по балансу энергии:

$$max|\frac{y_n^s - y_n^{s-1}}{y_n^s}| <= \varepsilon_1$$

для всех $n = 0, 1, ...N$. и

$$max|\frac{f_1^s - y_2^s}{f_1^s}| <= \varepsilon_1$$

где

$$f_1 = F_0 - \alpha(T(l) - T_0)$$

$$f_2 = 4n_p^2\sigma_0^1 k(T(x))(T^4(x) - T_0^4)dx$$

Исходный код алгоритмов

В листинге 1 представлена реализация алгоритма решения задачи. В листингах 2 – 5 приведены вспомогательные функции и главная программа.

Листинг 1 – Реализация алгоритма решения задачи

```
package optic

import (
    "fmt"
    "math"

    "gonum.org/v1/gonum/interp"
)

func GetRConds(lt, kt interp.AkimaSpline, tbl FArr64) Conds {
    var cs Conds

    cs.K = getXRight(lt, tbl, 0) +
        math.Pow(Params.H, 2)/8*getPRight(kt, tbl, 0) +
```

```go
        math.Pow(Params.H, 2)/4*getP(kt, tbl, 0)
    cs.M = math.Pow(Params.H, 2)/8*getPRight(kt, tbl, 0) - getXRight(lt, tbl, 0)
    cs.P = Params.H*Params.F0 + math.Pow(Params.H, 2)/4*(getFRight(kt, tbl, 0)+getFLeft(kt, tbl, 0))

    return cs
}

func GetLConds(lt, kt interp.AkimaSpline, tbl FArr64, n int) Conds {
    var cs Conds

    cs.K = getXLeft(lt, tbl, n)/Params.H -
        Params.Alpha - Params.H*getP(kt, tbl, n)/4 -
        Params.H*getPLeft(kt, tbl, n)/8
    cs.M = getXLeft(lt, tbl, n)/Params.H - Params.H*getPLeft(kt, tbl, n)/8
    cs.P = -(Params.Alpha*Params.T0 + (getFRight(kt, tbl, n)+getFLeft(kt, tbl, n))/4*Params.H)

    return cs
}

func A(lt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (lt.Predict(tbl[n]) + lt.Predict(tbl[n-1])) / 2 / Params.H
}

func B(lt, kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return A(lt, tbl, n) + C(lt, tbl, n) + getP(kt, tbl, n)*Params.H
}

func C(lt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (lt.Predict(tbl[n]) + lt.Predict(tbl[n+1])) / 2 / Params.H
}

func D(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return getF(kt, tbl, n) * Params.H
}

func Interpolate(xs, ys FArr64) interp.AkimaSpline {
    var as interp.AkimaSpline

    err := as.Fit(xs, ys)
    if err != nil {
        fmt.Println("Failed to initialize spline")
    }

    return as
}
```

```
func getP(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return 4 * Params.Np * Params.Np * Params.Sigma * kt.Predict(tbl[n]) *
        math.Pow(tbl[n], 3)
}

func getF(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return 4 * Params.Np * Params.Np * Params.Sigma * kt.Predict(tbl[n]) *
        math.Pow(Params.T0, 4)
}

func getXRight(lt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (lt.Predict(tbl[n]) + lt.Predict(tbl[n+1])) / 2
}

func getXLeft(lt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (lt.Predict(tbl[n]) + lt.Predict(tbl[n-1])) / 2
}

func getPRight(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (getP(kt, tbl, n) + getP(kt, tbl, n+1)) / 2
}

func getPLeft(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (getP(kt, tbl, n) + getP(kt, tbl, n-1)) / 2
}

func getFRight(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (getF(kt, tbl, n) + getF(kt, tbl, n+1)) / 2
}

func getFLeft(kt interp.AkimaSpline, tbl FArr64, n int) float64 {
    return (getF(kt, tbl, n) + getF(kt, tbl, n-1)) / 2
}
```

Листинг 2 – Реализация вспомогательных типов

```
package optic

// FArr64 is used to represent []float64
type FArr64 []float64

// FMat64 is used to represent [][]float64
type FMat64 []FArr64

// Conds is used to represent optic system conditions
type Conds struct {
    K float64
    M float64
    P float64
```

```go
}

// Optic is used to represent optic system parameters
type Optic struct {
    Np float64
    L float64
    T0 float64
    Tconst float64
    Sigma float64
    F0 float64
    Alpha float64
    H float64
}
```

Листинг 3 – Константы

```go
package optic

var (
    LambdaTbl = FMat64{
        FArr64{300, 500, 800, 1100, 2000, 2400},
        FArr64{1.36e-2, 1.63e-2, 1.81e-2, 1.98e-2, 2.50e-2, 2.74e-2},
    }
    KTbl = FMat64{
        FArr64{293, 1278, 1528, 1677, 2000, 2400},
        FArr64{2.0e-2, 5.0e-2, 7.8e-2, 1.0e-1, 1.3e-1, 2.0e-1},
    }
    Params = Optic{1.4, 0.2, 300, 400, 5.668e-12, 100, 0.05, 1e-4}
)
```

Листинг 4 – Реализация функций отрисовки графика

```go
package optic

import (
    "fmt"
    "image/color"
    "os"

    "gonum.org/v1/plot"
    "gonum.org/v1/plot/plotter"
    "gonum.org/v1/plot/vg"
)

// DrawPlot is used to draw plot with given coordinates and meta info
func DrawPlot(xs, ys FArr64, title, xl, yl, file string) {
    p := plot.New()

    p.Title.Text = title
```

```
    p.X.Label.Text = xl
    p.Y.Label.Text = yl
    p.Add(plotter.NewGrid())

    dots := convertDots(xs, ys)

    l, err := plotter.NewLine(dots)
    if err != nil {
        fmt.Println("Error:", err)
        os.Exit(1)
    }
    l.LineStyle.Width = vg.Points(1)
    l.LineStyle.Color = color.RGBA{B: 255, A: 255}

    p.Add(l)

    if err := p.Save(10*vg.Inch, 4*vg.Inch, file); err != nil {
        panic(err)
    }
}

func convertDots(xs, ys FArr64) plotter.XYs {
    var conv plotter.XYs

    for i := 0; i < len(xs); i++ {
        d := plotter.XY{
            X: xs[i],
            Y: ys[i],
        }
        conv = append(conv, d)
    }

    return conv
}
```

Листинг 5 – Главная программа

```
package main

import (
    "lab_03/optic"
)

func main() {
    lt := optic.Interpolate(optic.LambdaTbl[0], optic.LambdaTbl[1])
    kt := optic.Interpolate(optic.KTbl[0], optic.KTbl[1])
    tbl := make(optic.FArr64, int(1./optic.Params.H)+2)

    xil := optic.FArr64{0}
```

```
    etal := optic.FArr64{0}
    xl := optic.FArr64{}

    x := 0.
    n := 0

    for x+optic.Params.H < 1 {
        xl = append(xl, x)
        xil = append(xil, optic.C(lt, tbl, n)/(optic.B(lt, kt, tbl, n)-optic.A(lt, tbl,
            n)*xil[n]))
        etal = append(etal, (optic.D(kt, tbl, n)+optic.A(lt, tbl, n)*xil[n])/
            (optic.B(lt, kt, tbl, n)-optic.A(lt, tbl, n)*xil[n]))

        n++
        x += optic.Params.H
    }

    xl = append(xl, x+optic.Params.H, x+optic.Params.H*2)

    lcs := optic.GetLConds(lt, kt, tbl, n)
    tbl[n] = (lcs.P - lcs.M*xil[n]) / (lcs.K + lcs.M*xil[n])

    for i := n - 1; i > -1; i-- {
        tbl[i] = xil[i+1]*tbl[i+1] + etal[i+1]
    }

    optic.DrawPlot(xl, tbl, "T(x)", "x", "T", "data/tx.png")
}
```

Результат работы программы

Представить разностный аналог краевого условия при $x = l$ и его краткий вывод интегро-интерполяционным методом.

Для начала проинтегрируем уравнение на отрезке $[X_{n-\frac{1}{2}}; x_n]$:

$$-\int_{x_{n-\frac{1}{2}}}^{x_n} \frac{dF}{dx} dT - \int_{x_{n-\frac{1}{2}}}^{x_n} P(T) \cdot T^4 dT + \int_{x_{n-\frac{1}{2}}}^{x_n} f(t) dT = 0$$

Для получения второго и третьего интеграла воспользуемся методом трапеций:

$$F_{n-\frac{1}{2}} - F_n - \frac{h}{4}(p_n y_n + p_{n-\frac{1}{2}} y_{n-\frac{1}{2}}) + \frac{h}{4}(f_n + f_{n-\frac{1}{2}}) = 0$$

Учитывая, что:

$$F_{n-\frac{1}{2}} = x_{n-\frac{1}{2}} \frac{y_{n-1}}{y_n} h$$

$$F_n = \alpha_n (y_n - T_0)$$

$$y_{n-\frac{1}{2}} = \frac{y_n + y_{n-1}}{2h}$$

Имеем:

$$\frac{x_{n-\frac{1}{2}} y_{n-1}}{h} - \frac{x_{n-\frac{1}{2}} y_n}{h} - \alpha_n y_n + \alpha_n T_0 - \frac{h p_n y_n}{48} - \frac{h p_{n-\frac{1}{2}} y_n}{8} - \frac{h p_{n-\frac{1}{2}} y_{n-1}}{8} + \frac{f_{n-\frac{1}{2}} + f_n}{4} h = 0$$

$$y_n\left(-\frac{x_{n-\frac{1}{2}}}{h} - \alpha_n - \frac{h p_n}{4} - \frac{h p_n - \frac{1}{2}}{8}\right) + y_{n-1}\left(\frac{x_{n-\frac{1}{2}}}{h} - \frac{h p_{n-\frac{1}{2}}}{8}\right) = -\left(\alpha_n T_0 + \frac{f_n - \frac{1}{2}}{4} h\right)$$

График зависимости температуры $T(x)$ координаты x при заданных выше параметрах.

На рисунке 1 представлен график зависимости $T(x)$ при заданных выше параметрах.

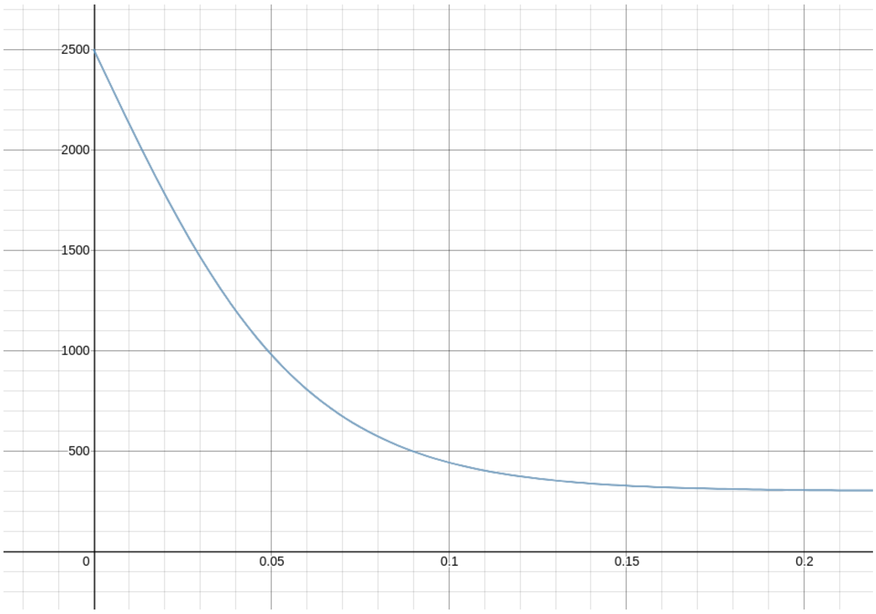

Рисунок 1 – График зависимости $T(x)$ при заданных выше параметрах

График зависимости $T(x)$ при $F_0 =$ -10 $\frac{\text{Вт}}{\text{см}^2}$.

На рисунке 2 представлен график зависимости $T(x)$ при $F_0 =$ -10 $\frac{\text{Вт}}{\text{см}^2}$.

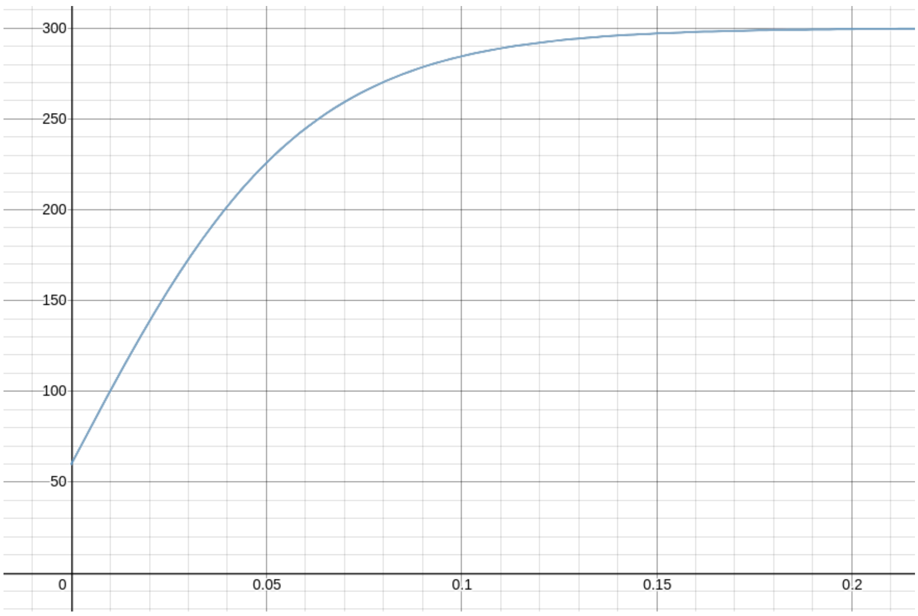

Рисунок 2 – График зависимости $T(x)$ при $F_0 =$ -10 $\frac{\text{Вт}}{\text{см}^2}$

График зависимости $T(x)$ при увеличенных значениях α (например, в 3 раза). Сравнить с п. 2.

На рисунке 3 представлен график зависимости $T(x)$ при увеличенных значениях α (голубая линия – увеличенное значение, синяя – значения из п. 2.).

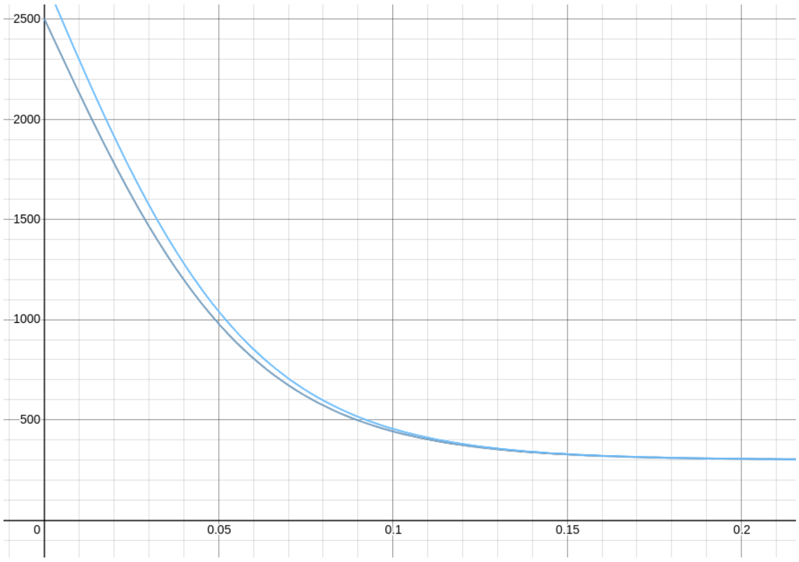

Рисунок 3 – График зависимости $T(x)$ при увеличенных значениях α

График зависимости $T(x)$ при $F_0 = 0$.

На рисунке 4 представлен график зависимости $T(x)$ при $F_0 = 0$.

Рисунок 4 – График зависимости $T(x)$ при $F_0 = 0$

Для указанного в задании исходного набора параметров привести данные по балансу энергии.

- Точность выхода $\varepsilon_1 = 0.064$ (по температуре);
- Точность выхода $\varepsilon_2 = 1.1$ (по балансу).

Ответы на контрольные вопросы

Вопрос 1. Какие способы тестирования программы можно предложить?

Ответ. В качестве еще одного способа тестирования можно проследить закономерности: при $F_0 > 0$ происходит охлаждение пластины, а при $F_0 < 0$ – нагревание. Кроме того, при увеличении показтеля теплосъема, уровень должен снижаться, а градиент увеличиваться.

Вопрос 2. Получите простейший разностный аналог нелинейного краевого условия при $x = l$.

Ответ. Для получения разностного аналога необходимо изначально аппроксимировать производную:

$$\frac{dT}{dx} = \frac{y_N - y_{N-1}}{h}$$

Подставим полученное выражение в исходное уравнение:

$$-k_N \frac{y_n - y_{N-1}}{h} = \alpha_N(y_N - T_0) + \varphi(y_N)$$

Учтём, что $y_{N-1} = \xi_N y_N + \eta_N$:

$$-k_N(y_N - \xi_N y_N + \eta_N) = \alpha_N(y_N - T_0)h + \varphi(y_N)h$$

Приведя подобные слагаемые, получим:

$$\varphi(y_N)h + (k_N + \alpha_N h - k_N\xi_N - k_N\eta_N)y_N - h\alpha_N T_0 = 0$$

Вопрос 3. Опишите алгоритм применения метода прогонки, если при $x = 0$ краевое условие квазилинейное (как в настоящей работе), а при $x = l$, как в п. 2.

Ответ. Для начала найдем начальные прогочные коэффициенты по формулам (коэффициенты M_0, P_0, K_0 известны):

$$\xi = \frac{-M_0}{P_0}$$

$$\eta = \frac{-K_0}{P_0}$$

Далее, находим последующие прогоночные коэффициенты:

$$\xi_{n+1} = \frac{C_n}{B_n - A_n\xi_n}$$

$$\eta_{n+1} = \frac{F_n + A_n\eta_n}{B_n - A_n\xi_n}$$

Из уравнения, полученного в п. 2., можем получить y_N. По прогочной формуле можем найти все значения неизвестных y_N

$$y_n = \xi_{n+1}y_{n+1} + \eta_{n+1}$$

Вопрос 4. Опишите алгоритм определения единственного значения сеточной функции y_p в одной заданной точке p. Использовать встречную прогонку, т.е. комбинацию правой и левой прогонок.

Ответ. Вычислим начальные прогоночные коэффициенты.

Для правой прогонки:

$$\xi = \frac{-M_0}{P_0}$$

$$\xi = \frac{-K_0}{P_0}$$

Для левой прогонки:

$$\alpha_{N-1} = \frac{-M_N}{K_N}$$

$$\beta_{N-1} = \frac{-P_N}{K_N}$$

Найдем прогоночные коэффициенты.

Для левой прогонки:

$$\xi_{n+1} = \frac{C_n}{B_n - A_n\xi_n}$$

$$\eta_{n+1} = \frac{F_n + A_n\eta_n}{B_n - A_n\xi_n}$$

Для правой прогонки:

$$\alpha_{n-1} = \frac{A_n}{B_n - C_n \alpha_n}$$

$$\beta_{n-1} = \frac{F_n + C_n \beta_n}{B_n - C_n \alpha_n}$$

Левые и правые прогонки:

$$y_n = \xi_{n+1} y_{n+1} + \eta_{n+1}$$

$$y_n = \alpha_{n-1} y_{n+1} + \beta_{n-1}$$

Выразим y_p:

$$y_{p-1} = \xi_p y_p + \eta_p$$

$$y_p = \alpha_{p-1} y_{p-1} + \beta_{p-1}$$

$$y_p = \frac{\xi_{n+1} \beta_n + \eta_{n+1}}{1 - \xi_{n+1} \alpha_n}$$